МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «Уральский государственный экономический университет»

В.П. Часовских

АИС-7 «Разработка ИС и средств администрирования»

Екатеринбург 2024

Лабораторная работа 7. Для выбранной модели создание проекта ИС средствами технологии объектного проектирования информационных систем ASP.NET Core MVC, формирование контроллеров и программ обработки модели, создание базы данных SQL.

МОДЕЛИ

Желтым выделено поле для изменения. Указывается выбранное имя проекта и папка модеи

1. Преподователь

```
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Web.Mvc;


namespace KafedraUGLTU.Domain.Entities
{
    public class Prepod
    {
        [HiddenInput(DisplayValue = false)]
        public int PrepodID { get; set; }

        [DisplayName("Назвнияе института/факультета")]

        [Required(ErrorMessage = "Укажите название института/факультета")]
        public string Institut  { get; set; }

        [DisplayName("Аббревиатура назвния института/факультета")]
        public string InstitutAbriviatura  { get; set; }


        [DisplayName("Назвние кафедры")]
        public string Kafedra   { get; set; }

        [DisplayName("Аббревиатура назвния кафедры")]
        public string KafedraAbriviatura  { get; set; }

         [DisplayName("Фамилия")]
         [Required(ErrorMessage = "Укажите фамилию")]
         public string Familiy { get; set; }

        [DisplayName("Имя")]
        [Required(ErrorMessage = "Укажите имя")]
        public string Imy { get; set; }

        [DisplayName("Отчество")]
        [Required(ErrorMessage = "Укажите отчество")]
        public string Otchestvo { get; set; }

        [DisplayName("Год рождения")]
        public string GodRogd { get; set; }

        [DisplayName("Должность")]
        public string Dolgnost  { get; set; }

        [DisplayName("Ученая степень ВАК")]
        public string Stepen  { get; set; }

        [DisplayName("Ученое звание ВАК")]
        public string Zvanie  { get; set; }
```

```
        [DisplayName("Образование")]
        public string Obrazovanie { get; set; }

        [DisplayName("Читаемые дисциплины")]
        public string ChitDiszip { get; set; }

        [DisplayName("Опыт работы")]
        public string OpitRab { get; set; }

        [DisplayName("Повышение квалификации за последние 5 лет")]
        public string PovKval { get; set; }

        [DisplayName("Общий стаж работы")]
        public string StagRab { get; set; }

        [DisplayName("Научно-педагогический стаж работы")]
        public string StagPRab { get; set; }

        [DisplayName("Адрес электронной почты; ")]
        public string Email { get; set; }


        [DisplayName("Ссылка на фото")]
        public string Foto { get; set; }


        public string FullName
        {
            get { return Familiy + " " + Imy + " " + Otchestvo; }
        }

        public virtual ICollection<Prepod> Prepods { get; set; }

    }
}
```

2. Публикация

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;


namespace KafedraUGLTU.Domain.Entities
{
    public class Publikazii
    {
        public int PublikaziiID { get; set; }
        [Display(Name = "Тип литературы")]
        public string TipUchNauchLiteraturiID { get; set; }
        [Display(Name = "Кафедра")]
        public int UrovenPublikaziiID { get; set; }

        [Display(Name = "Название")]
        [Required(ErrorMessage = "Укажите название публикации")]
        public string Nazvanie { get; set; }

        [Display(Name = "Авторы")]
        [Required(ErrorMessage = "Укажите авторов публикациии")]
        public string Avtor { get; set; }
        [Display(Name = "Издание")]
        [Required(ErrorMessage = "Укажите издание")]
        public string Izdanie { get; set; }
```

```
        [Display(Name = "Год")]
        [Required(ErrorMessage = "Укажите год публикации")]
        public string God { get; set; }
        [Display(Name = "Файл загрузки копии публикации")]
        [Required(ErrorMessage = "Укажите файл копии публикации")]
        public string Zagruzit { get; set; }


        [Display(Name = "Год загрузки копии публикации")]
        public string GodZagruzit { get; set; }

        [Display(Name = "Файл загрузки скриншота из БД  публикации")]

        public string Publikazii01 { get; set; }


    }
}
```

3. Темы контрольных работ

```
using KafedraUGLTU.Domain.Entities;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace KafedraUGLTU.Domain.Entities
{
    public class TemiKontRab
    {

        public int TemiKontRabID { get; set; }

        [DisplayName("Тема(задача) контрольной работы")]
        public string KontRabotNazvanie { get; set; }

        [DisplayName("Тип работы - для магистров начинется с МАГ_тип работы")]
        //  string s = "Hello, World"; s.Substring(1,3));    получаем строку начиная с 1
позиции длиной 3 симола, получим "Wor"
         public int _0TipRabotiID { get; set; }

        [DisplayName("Студент")]
         public string Student { get; set; }

        [DisplayName("Дисциплина")]
         public string Disziplina  { get; set; }


        [DataType(DataType.MultilineText)]
        [Display(Name = "Конт работа")]
        [MaxLength(2500, ErrorMessage = "В рецензии можно указать до  2500 символов.")]
        public string KontDopPole01 { get; set; }


         [DataType(DataType.MultilineText)]
         [Display(Name = "Рецензия на контрольную работу преподавателя")]
         [MaxLength(2500, ErrorMessage = "В рецензии можно указать до  2500 символов.")]
        public string KuKontrDopPole05 { get; set; }

        ublic virtual ICollection<TemiKontRab> TemiKontRabs { get; set; }
    }
}
```

4. Учеба студента

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;


namespace KafedraUGLTU.Domain.Entities
{
  public  class UchebaStudenta
    {


         [Display(Name = "Студент")]
        public string Student  { get; set; }


        [Display(Name = "Дисциплина")]
         public string Disziplina  { get; set; }



       [Display(Name = "Курсовая работа оценка")]
        public string KursRabOzenka { get; set; }

        [Display(Name = "Контрольная работа ДА - НЕТ")]
        [UIHint("YesNo")]
        public string YesNo1 { get; set; }

        [Display(Name = "Контрольная работа оценка")]
        public string KontRabOzenka { get; set; }

        [Display(Name = "Лабораторно-практические занятия ДА - НЕТ")]
        [UIHint("YesNo")]
        public string YesNo2 { get; set; }
        [Display(Name = "Число лабораторно-практических занятий")]
        public string ChisloLabPrakt { get; set; }
        [Display(Name = "Лабораторно-практические занятия оценка")]
        public string LabPraktOzenka { get; set; }

        [Display(Name = "Самостоятельная  работа ДА - НЕТ")]
        [UIHint("YesNo")]
        public string YesNo3 { get; set; }
        [Display(Name = "Тема самостоятельной работы")]
        public string TemiSamRab { get; set; }
        [Display(Name = "Самостоятельная работа оценка")]
        public string SamRabOzenka { get; set; }

        [Display(Name = "Зачет ДА - НЕТ")]
        [UIHint("YesNo")]
        public string YesNo4 { get; set; }
        [Display(Name = "Оценка ")]
        public string ZachetOzenka { get; set; }

        [Display(Name = "Экзамен ДА - НЕТ")]
        [UIHint("YesNo")]
        public string YesNo5 { get; set; }
        [Display(Name = "Оценка ")]
        public string EkzamentOzenka { get; set; }
        [Display(Name = "Зачетных единиц ")]
        public string ZachetEdiniz { get; set; }
```

```
        [Display(Name = "Ссылка на работу и справку о плагиате")]
        public string UchebaPole01 { get; set; }
        [Display(Name = "Ссылка на протокол защиты")]
        public string UchebaPole02 { get; set; }
        [Display(Name = "Приобретённые компетенции, записываются через точку с запятой
без пробелов")]

        public virtual ICollection<UchebaStudenta> UchebaStudentas { get; set; }


    }
}
```

5. Вопрос – ответ

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Web.Mvc;

namespace KafedraUGLTU.Domain.Entities
{
    public class VoprosOtvet
    {

        [HiddenInput(DisplayValue = false)]
        [Display(Name = "Ключ записи Вопрос - Ответ")]
        public int VoprosOtvetID { get; set; }
        [Display(Name = "Номер студенческого билета  ")]
        [Required(ErrorMessage = "Пожалуйста  укажите номер студенческого билета.")]
        public string NomerStudBileta { get; set; }

        [DataType(DataType.MultilineText)]
        [Required(ErrorMessage = "Пожалуйста  сформулируйте свой вопрос.")]
        [Display(Name = "Вопрос")]
        [MaxLength(500, ErrorMessage = "В вопросе можно указать до  500 символов.")]
        public string Vopros { get; set; }
        [Display(Name = "Ответ")]
        [DataType(DataType.MultilineText)]
        public string Otvet { get; set; }

        [Display(Name = "Дата   в формате - дд.мм.гггг ")]
        [DisplayFormat(DataFormatString = "{0:d}", ApplyFormatInEditMode = true)]  //***
отображается только дата, без времени ApplyFormatInEditMode указывает, что данное
форматирование должно применяться также для значений, отображающихся в текстовых строках,
предназначенных для редактирования
        [Required(ErrorMessage = "Пожалуйста  укажите дату в формате дд-мм-гггг.")]
        public DateTime DateAdded { get; set; }

        [Display(Name = "Если вопрос корректен, то указать Да")]
        public string DR1 { get; set; }



        public virtual ICollection<VoprosOtvet> VoprosOtvets { get; set; }


    }



}
```

Разработка проекта ИС основывается на технологии объектного проектирования ASP.NET Core MVC.

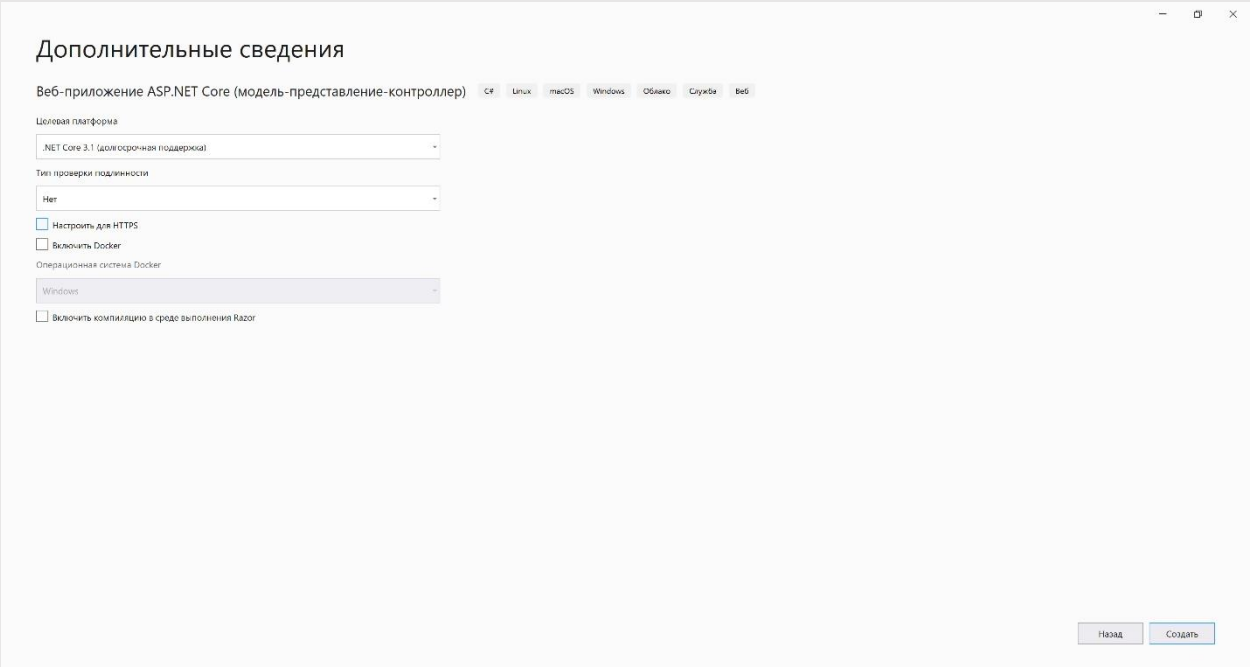
В процессе выполнения работы Вы научитесь:

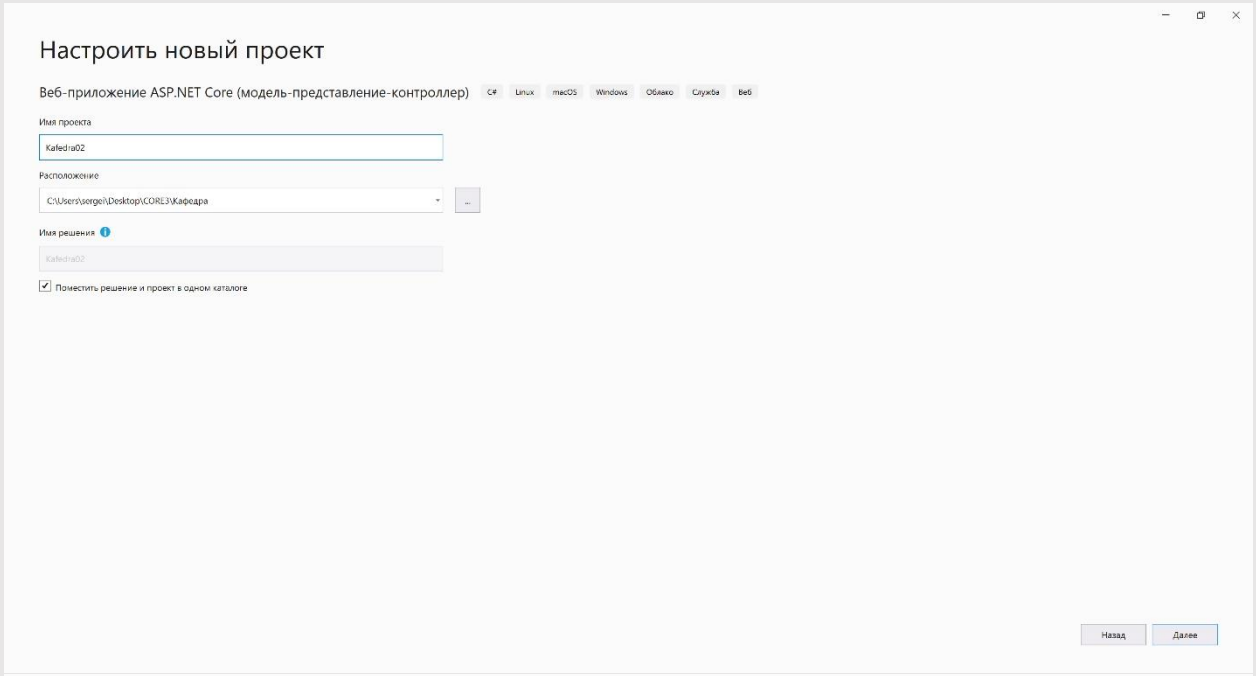
- Создавать веб-приложение и средства администрирования.
- Добавление модели и формирование шаблона.
- Работать с базой данных.
- Добавление поиска и проверки.

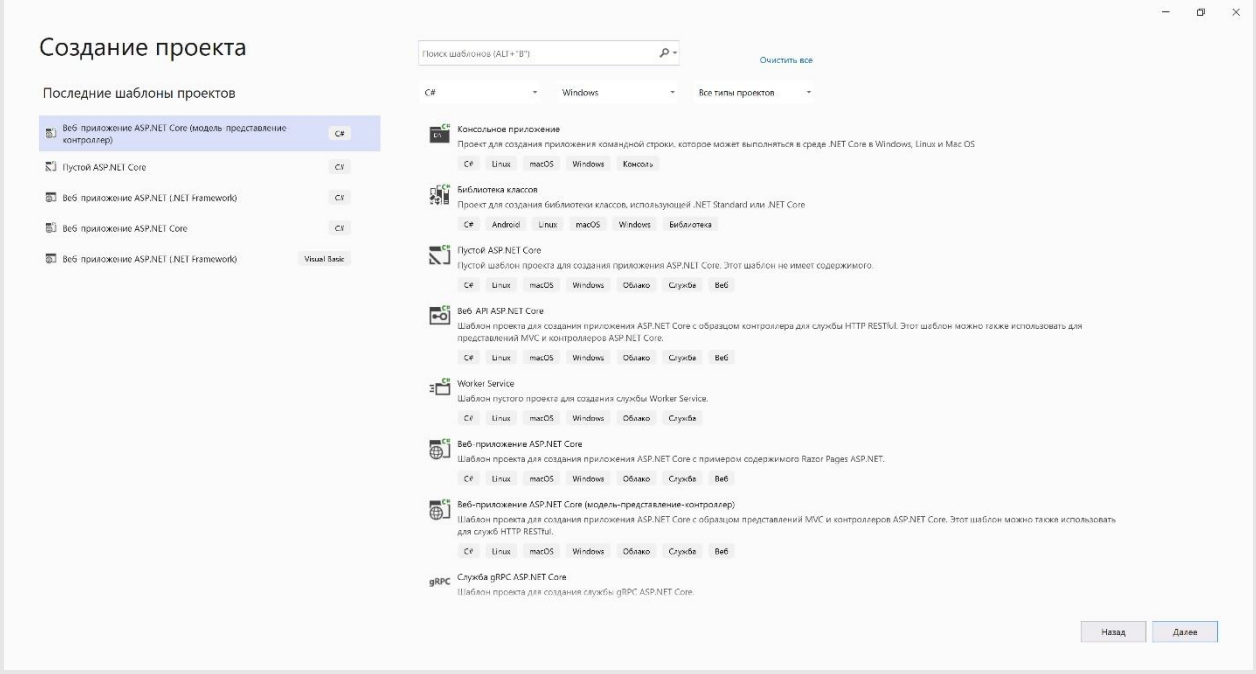
Необходимая среда для выполнения работы

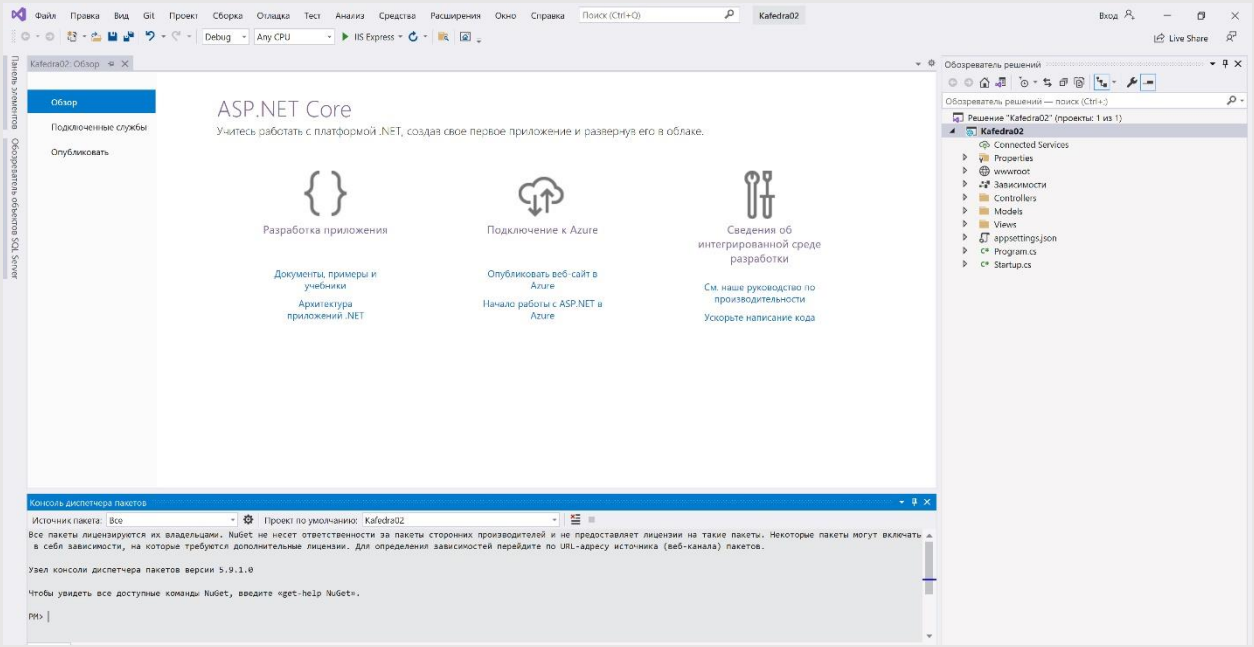
- Visual Studio 2019 16.4 или более поздней версии с рабочей нагрузкой **ASP.NET и разработка веб-приложений**
- Пакет SDK для .NET Core 3.1 или более поздней версии

Необходимо запустить Visual Studio 2019 и последовательно выполнить последующие действия.




Настроить новый проект
Веб-приложение ASP.NET Core (модель-представление-контроллер)
C# Linux macOS Windows Облако Служба Веб
Имя проекта
Kafedra02
Расположение
C:\Users\serge\Desktop\CORE3\Кафедра
...
Имя решения
Kafedra02
Поместить решение и проект в одном каталоге
Назад
Далее


Создание проекта
Последние шаблоны проектов
Веб-приложение ASP.NET Core (модель представление контроллер) C#
Пустой ASP.NET Core C#
Веб-приложение ASP.NET (.NET Framework) C#
Веб-приложение ASP.NET Core C#
Веб-приложение ASP.NET (.NET Framework) Visual Basic
Поиск шаблонов (ALT+"B")
Очистить все
C#
Windows
Все типы проектов
Консольное приложение
Проект для создания приложения командной строки, которое может выполняться в среде .NET Core в Windows, Linux и Mac OS
C# Linux macOS Windows Консоль
Библиотека классов
Проект для создания библиотеки классов, использующей .NET Standard или .NET Core
C# Android Linux macOS Windows Библиотека
Пустой ASP.NET Core
Пустой шаблон проекта для создания приложения ASP.NET Core. Этот шаблон не имеет содержимого.
C# Linux macOS Windows Облако Служба Веб
Веб-API ASP.NET Core
Шаблон проекта для создания приложения ASP.NET Core с образцом контроллера для службы HTTP RESTful. Этот шаблон можно также использовать для представлений MVC и контроллеров ASP.NET Core.
C# Linux macOS Windows Облако Служба Веб
Worker Service
Шаблон пустого проекта для создания службы Worker Service.
C# Linux macOS Windows Облако Служба
Веб-приложение ASP.NET Core
Шаблон проекта для создания приложения ASP.NET Core с примером содержимого Razor Pages ASP.NET.
C# Linux macOS Windows Облако Служба Веб
Веб-приложение ASP.NET Core (модель-представление-контроллер)
Шаблон проекта для создания приложения ASP.NET Core с образцом представлений MVC и контроллеров ASP.NET Core. Этот шаблон можно также использовать для служб HTTP RESTful.
C# Linux macOS Windows Облако Служба Веб
gRPC
Служба gRPC ASP.NET Core
Шаблон проекта для создания службы gRPC ASP.NET Core.
Назад
Далее



Обратите внимание на «Настроить для HTTPS» - квадратик пуст.

Настройка стиля визуализации

Выполните незначительную настройку меню, макета и домашней страницы сайта.

Откройте файл *Views/Shared/_Layout.cshtml* и внесите следующие изменения:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Kafedra02</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
```

```
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white
border-bottom box-shadow mb-3">
            <div class="container">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">Kafedra02</a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-
reverse">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
controller="Home" asp-action="Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
controller="Home" asp-action="Privacy">Privacy</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2023 - Kafedra02 - <a asp-area="" asp-controller="Home" asp-
action="Privacy">Privacy</a>
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @RenderSection("Scripts", required: false)
</body>
</html>
```

Замените содержимое файла *Views/Home/Index.cshtml* следующим кодом, который заменяет текст о ASP.NET и MVC описанием этого приложения:

```
  ViewBag.Title = "Home Page";

 <h1 style="text-align:center; color:#796310">Дисциплина « Современные технологии
разработки программного обеспечения» в среде ASP.NET Core MVC </h1>
```

Создадим в папке Models сущностей (таблицу будущей базы данных) – тип файла класс Disziplina

И заменим сгенерированный код следующим:

```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
namespace Kafedra02
{
    public  class Disziplina
    {
        [HiddenInput(DisplayValue = false)]
        [Display(Name = "Ключ записи")]
        public int DisziplinaID { get; set; }
        [Display(Name = "Название дисциплины")]
        public string Nazvanie { get; set; }
        [Display(Name = "Зачетные единицы")]
        public string ZachetEdin { get; set; }
        [Display(Name = "Курсовая работа")]
        public string KursRabota { get; set; }
        [Display(Name = "Контрольная работа")]
        public string KontRabota { get; set; }
        [Display(Name = "Форма получения оценки")]
        public string FormaOzenki { get; set; }
        [Display(Name = "Ссылка на рабочую  программу в папке Uploads/RabProg")]
        public string ObrzavProgramma { get; set; }
        [Display(Name = "Имя файла рабочей программы")]
        public string DDopPole01 { get; set; }
        [Display(Name = "Читает кафедра")]
        public string DDopPole02 { get; set; }
        [Display(Name = "Направление- Бакалавр или Магистр или Аспирант")]
        public string DDopPole03 { get; set; }
        [Display(Name = "Приобретаемые компетенции, записываются через точку с запятой
без пробелов")]
        public string DDopPole04 { get; set; }
        [Display(Name = "Доп. поле 05")]
        public string DDopPole05 { get; set; }



        public virtual ICollection<Disziplina> Disziplinas { get; set; }

    }
}
}
```

Создание контекста базы данных

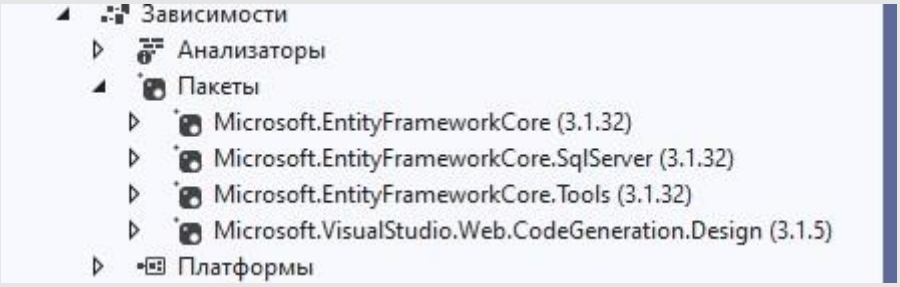
Контекст базы данных — это основной класс, который координирует функциональные возможности Entity Framework для заданной модели данных. Этот класс создается путем наследования от класса `Microsoft.EntityFrameworkCore.DbContext`. В коде указываются сущности, которые включаются в модель данных. Также вы можете настроить реакцию платформы Entity Framework на некоторые события. В этом проекте соответствующий класс называется `Kafedra02Context`.

В папке проекта создайте папку *Data*.

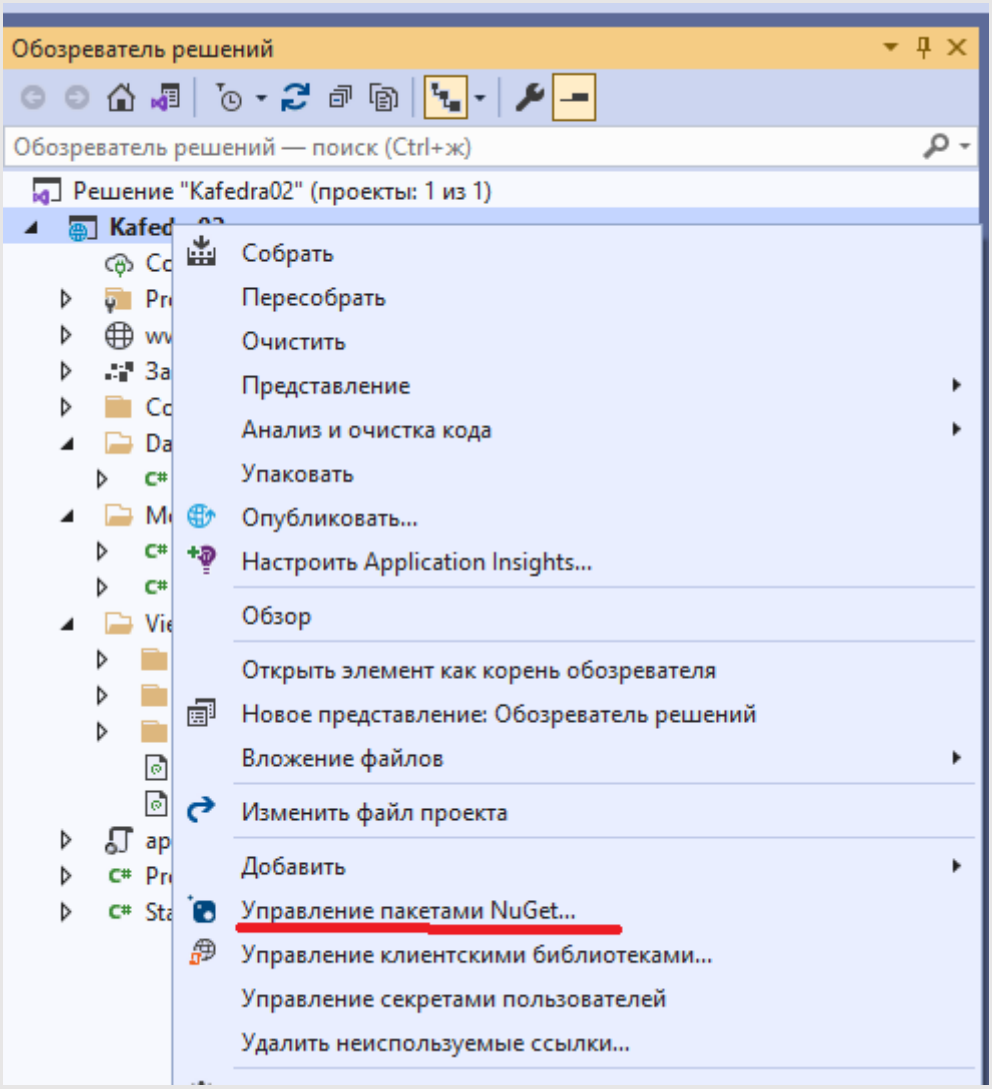
В папке *Data* создайте новый файл класса с именем *Kafedra02Context.cs* и замените код шаблона следующим кодом:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using Kafedra02;

namespace Kafedra02.Data
{
    public class Kafedra02Context : DbContext
    {
        public Kafedra02Context (DbContextOptions<Kafedra02Context> options)
            : base(options)
        {
        }

        public DbSet<Kafedra02.Disziplina> Disziplina { get; set; }
    }
}
```

Далее необходимо подключить необходимые программы в проект. Из скриншота видно, что должно быть доступно:



Подключение обеспечивается средством **«Управление пакетами NuGet…»**



Регистрация Kafedra02Context

ASP.NET Core по умолчанию реализует технологию внедрения зависимостей. С помощью внедрения зависимостей службы (например, контекст базы данных EF) регистрируются во время запуска приложения. Затем компоненты, которые используют эти службы (например, контроллеры MVC), обращаются к ним через параметры конструктора. Код конструктора контроллера, который получает экземпляр контекста, будет приведен позднее в этом учебнике.
Чтобы зарегистрировать `Kafedra02Context` как службу, откройте файл *Startup.cs* и добавьте выделенные строки в метод `ConfigureServices`.

```
using Kafedra02.Data;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
```

```
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Kafedra02
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the
container.



        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllersWithViews();



            services.AddDbContext<Kafedra02Context>(options =>

options.UseSqlServer(Configuration.GetConnectionString("Kafedra02Context")));
        }




        // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
            }
            app.UseStaticFiles();

            app.UseRouting();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }
}
```

Откройте файл *appsettings.json* и добавьте строку подключения, как показано в следующем примере.

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "Kafedra02Context": "Server=(localdb)\\mssqllocaldb;Database=KafedraContext-
1;Trusted_Connection=True;MultipleActiveResultSets=true",

  }
}
```

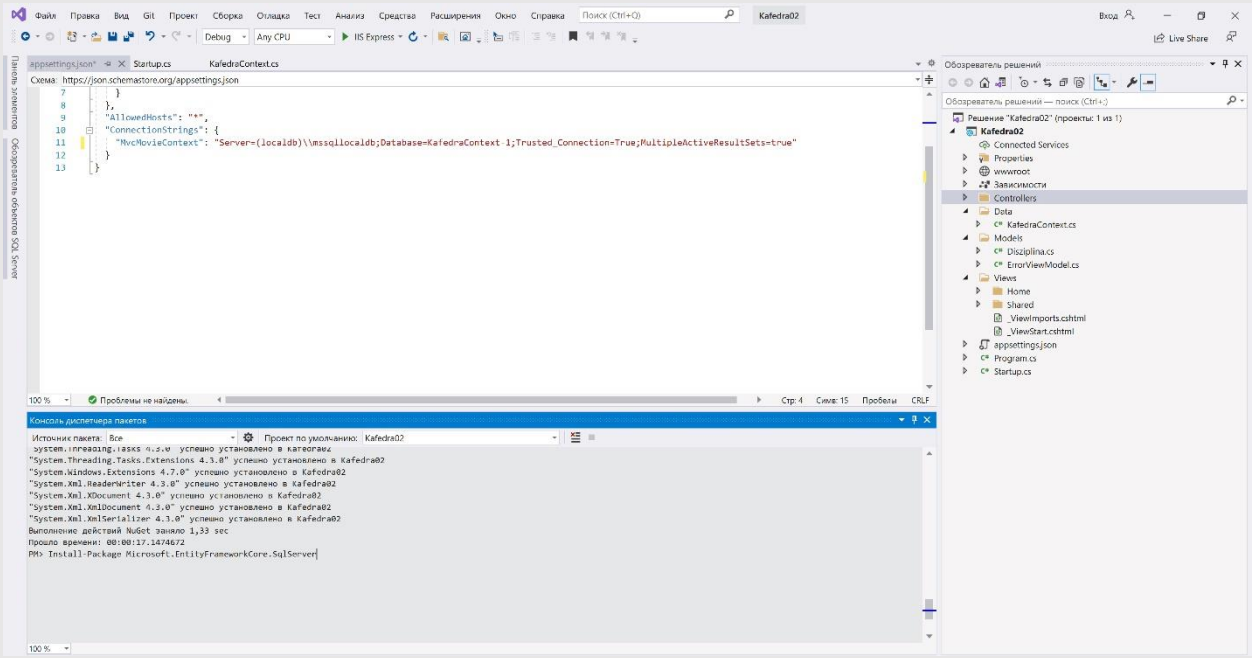
SQL Server Express LocalDB

Строка подключения указывает на базу данных SQL Server LocalDB. LocalDB — это упрощенная версия ядра СУБД SQL Server Express, предназначенная для разработки приложений и не ориентированная на использование в производственной среде. LocalDB запускается по запросу в пользовательском режиме, поэтому настройки не слишком сложны.

Формирования шаблона страниц дисциплины

Используйте средство формирования шаблонов, чтобы создать страницы для операций создания, чтения, обновления и удаления (CRUD) для модели дисциплина.

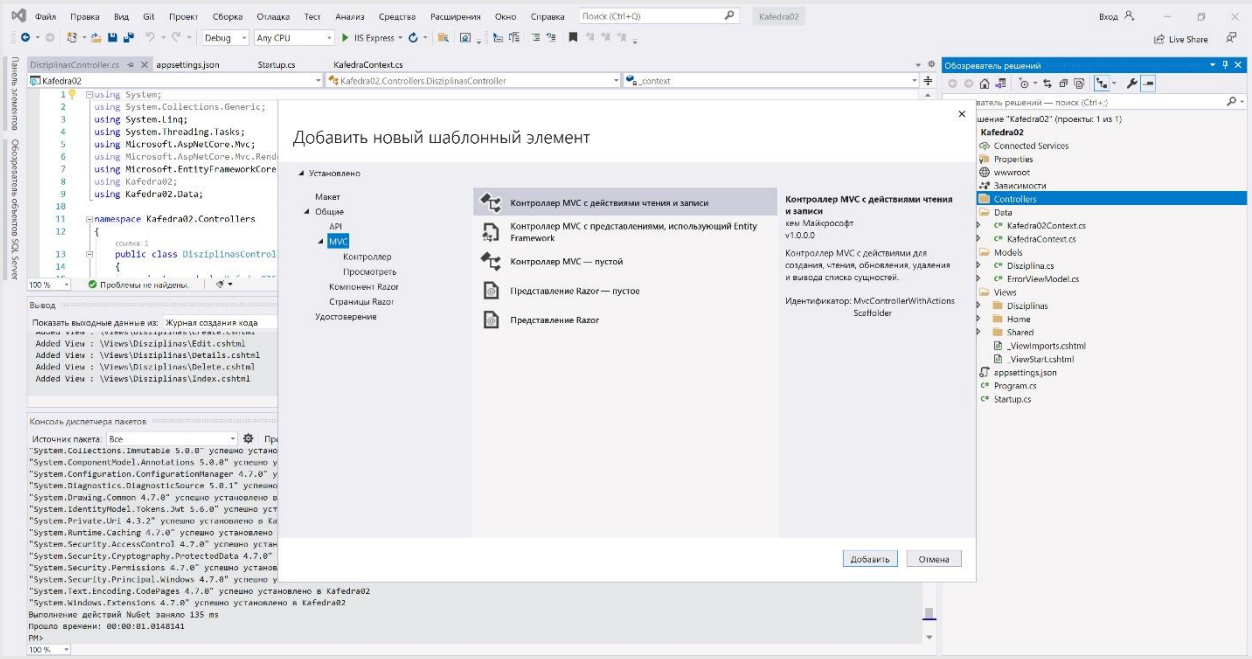
Для работы с базой данных используется пакет
Microsoft.EntityFrameworkCore.SqlServer

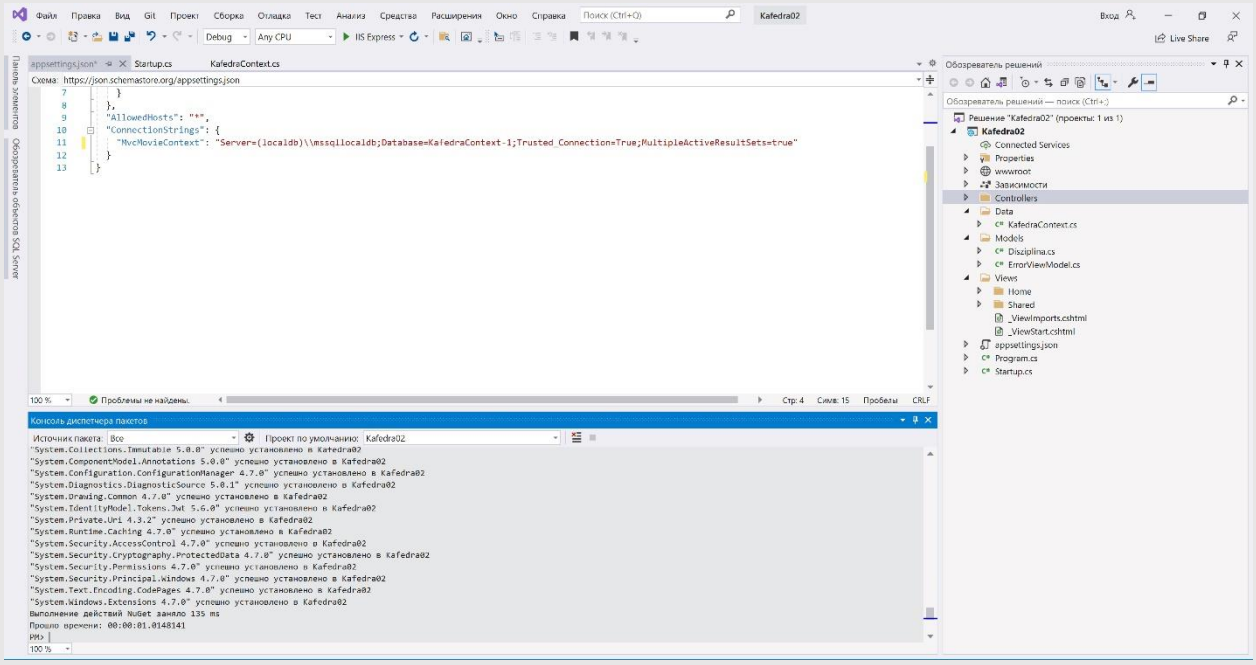


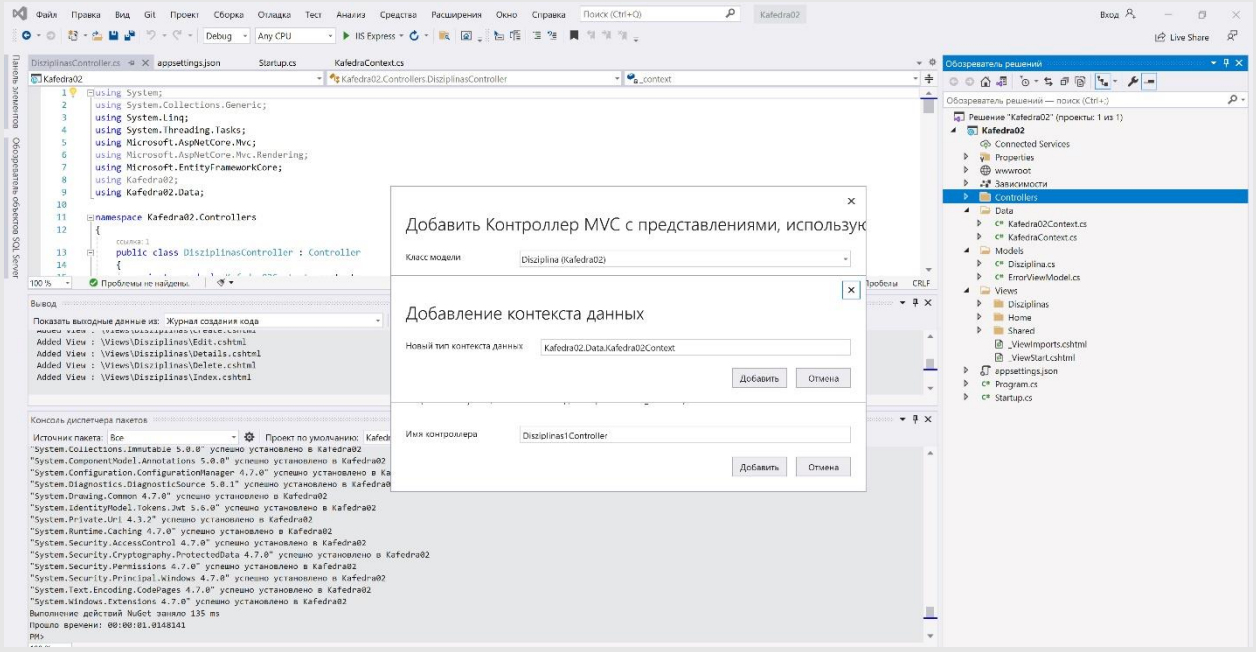
Приведенная выше команда добавляет поставщик EF Core для SQL Server. Пакет поставщика устанавливает пакет EF Core в качестве зависимости. Дополнительные пакеты устанавливаются автоматически на этапе формирования шаблонов далее в этом руководстве.

Формирования шаблона страниц дисциплин

Используйте средство формирования шаблонов, чтобы создать страницы для операций создания, чтения, обновления и удаления (CRUD) для модели фильма. Используется функция «***Контроллер NVC с представлением, использующей Entity Framework***».







Visual Studio создаст следующие компоненты:

- контроллер дисциплин (*Controllers/DisziplinasController.cs*);
- файлы представления для страниц Create, Delete, Details, Edit и Index (*Views/Disziplinas/*.cshtml*).

Автоматическое создание этих файлов называется *формированием шаблонов*.

Сформированные страницы пока использовать нельзя, так как база данных не существует. Если запустить приложение и щелкнуть ссылку **Дисциплина** в меню, появится сообщение об ошибке. *Невозможно открыть базу данных* или *отсутствует таблица*. Необходимо создать базу данных.

Первоначальная миграция – средства администрирования

Создайте базу данных с помощью функции миграций EF Core. Миграции — это набор средств, позволяющих создавать и обновлять базы данных в соответствии с моделью данных.

В меню **Сервис** последовательно выберите пункты **Диспетчер пакетов NuGet** > **Консоль диспетчера пакетов**

Введите следующие команды:

```
Add-Migration InitialCreate
Update-Database
```

- **`Add-Migration InitialCreate`**. Создает файл миграции *Migrations/{метка_времени}_InitialCreate.cs*. Аргумент `InitialCreate` — это имя миграции. Можно использовать любое имя, но по соглашению выбирается имя, которое описывает миграцию. Так как это первая миграция, созданный класс содержит код для создания схемы базы данных. Схема базы данных создается на основе модели, указанной в классе `Kafedra02Context`.
- **`Update-Database`**. Обновляет базу данных до последней миграции, созданной предыдущей командой.

Лабораторная работа 7-2. Формирование средств администрирования и удобных для пользователя форм просмотра, ввода, коррекции и удаления записей модели на русском языке. Ввод 5-7 записей модели.

Лабораторная работа 7-3. Изменение программы визуализации модели для пользователя:

- функция поиска записи по критерию;
- функция сортировки записей по значениям столбца;
- адаптация таблицы к размеру экрана визуализации.

Изменение **контроллера** модели «Дисциплина»

```
namespace Kafedra02.Controllers
{
    public class DisziplinasController : Controller
    {
        private readonly Kafedra02Context _context;

        public DisziplinasController(Kafedra02Context context)
        {
            _context = context;
        }
```

```
        public async Task<IActionResult> Index(string sortOrder, string searchString)
        {
            ViewData["NameSortParm"] = String.IsNullOrEmpty(sortOrder) ? "name_desc" :
"";
            ViewData["DateSortParm"] = sortOrder == "Date" ? "date_desc" : "Date";
            ViewData["CurrentFilter"] = searchString;


            var students = from s in _context.Disziplina
                           select s;

            if (!String.IsNullOrEmpty(searchString))
            {
                students = students.Where(s => s.Nazvanie.Contains(searchString)
                                       || s.KursRabota.Contains(searchString));
            }


            switch (sortOrder)
            {
                case "name_desc":
                    students = students.OrderByDescending(s => s.Nazvanie);
                    break;
                case "Date":
                    students = students.OrderBy(s => s.KursRabota);
                    break;
                case "date_desc":
                    students = students.OrderByDescending(s => s.KursRabota);
                    break;
                default:
                    students = students.OrderBy(s => s.Nazvanie);
                    break;
            }
            return View(await students.AsNoTracking().ToListAsync());
        }
```

```
        // GET: Disziplinas/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var disziplina = await _context.Disziplina
                .FirstOrDefaultAsync(m => m.DisziplinaID == id);
            if (disziplina == null)
            {
                return NotFound();
            }

            return View(disziplina);
        }

        // GET: Disziplinas/Create
        public IActionResult Create()
        {
```

```
            return View();
        }

        // POST: Disziplinas/Create
        // To protect from overposting attacks, enable the specific properties you want
to bind to, for
        // more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
Create([Bind("DisziplinaID,Nazvanie,ZachetEdin,KursRabota,KontRabota,FormaOzenki,ObrzavPr
ogramma,DDopPole01,DDopPole02,DDopPole03,DDopPole04,DDopPole05")] Disziplina disziplina)
        {
            if (ModelState.IsValid)
            {
                _context.Add(disziplina);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(disziplina);
        }

        // GET: Disziplinas/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var disziplina = await _context.Disziplina.FindAsync(id);
            if (disziplina == null)
            {
                return NotFound();
            }
            return View(disziplina);
        }

        // POST: Disziplinas/Edit/5
        // To protect from overposting attacks, enable the specific properties you want
to bind to, for
        // more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id,
[Bind("DisziplinaID,Nazvanie,ZachetEdin,KursRabota,KontRabota,FormaOzenki,ObrzavProgramma
,DDopPole01,DDopPole02,DDopPole03,DDopPole04,DDopPole05")] Disziplina disziplina)
        {
            if (id != disziplina.DisziplinaID)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(disziplina);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!DisziplinaExists(disziplina.DisziplinaID))
                    {
                        return NotFound();
```

```
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        return View(disziplina);
    }

    // GET: Disziplinas/Delete/5
    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        var disziplina = await _context.Disziplina
            .FirstOrDefaultAsync(m => m.DisziplinaID == id);
        if (disziplina == null)
        {
            return NotFound();
        }

        return View(disziplina);
    }

    // POST: Disziplinas/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        var disziplina = await _context.Disziplina.FindAsync(id);
        _context.Disziplina.Remove(disziplina);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool DisziplinaExists(int id)
    {
        return _context.Disziplina.Any(e => e.DisziplinaID == id);
    }
}
}
```

Изменение **метода “Index”** модели «Дисциплина»

```
@model IEnumerable<Kafedra02.Disziplina>

@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-action="Create">Create New</a>
</p>
```

```
<form asp-action="Index" method="get">
    <div class="form-actions no-color">
        <p>
            Find by name: <input type="text" name="SearchString"
value="@ViewData["CurrentFilter"]" />
            <input type="submit" value="Search" class="btn btn-default" /> |
            <a asp-action="Index">Back to Full List</a>
        </p>
    </div>
</form>


<table class="table">
    <thead>
        <tr>
            <th>


                <a asp-action="Index" asp-route-
sortOrder="@ViewData["NameSortParm"]">@Html.DisplayNameFor(model => model.Nazvanie)</a>
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ZachetEdin)
            </th>
            <th>


                <a asp-action="Index" asp-route-
sortOrder="@ViewData["DateSortParm"]">@Html.DisplayNameFor(model => model.KursRabota)</a>
            </th>
            <th>
                @Html.DisplayNameFor(model => model.KontRabota)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.FormaOzenki)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ObrzavProgramma)
            </th>

            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Nazvanie)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.ZachetEdin)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.KursRabota)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.KontRabota)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.FormaOzenki)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.ObrzavProgramma)
                </td>
```

```
                <td>
                    <a asp-action="Edit" asp-route-id="@item.DisziplinaID">Edit</a> |
                    <a asp-action="Details" asp-route-id="@item.DisziplinaID">Details</a>
|
                    <a asp-action="Delete" asp-route-id="@item.DisziplinaID">Delete</a>
                </td>
            </tr>
        }
    </tbody>
</table>
```